CS 1680
Intra-domain Routing

TODAY
- DHCP
- ROUTING: RIP

IP MILESTONE MEETINGS: HAPPENING NOW!

IP NEXT STEPS
- GEARUP #: RECORDING POSTED
- IMPLEMENTATION START GUIDE: USE THIS!

=> DON'T LEAVE PROJECT UNTIL THE LAST MINUTE

HW1: DUE TONIGHT

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti

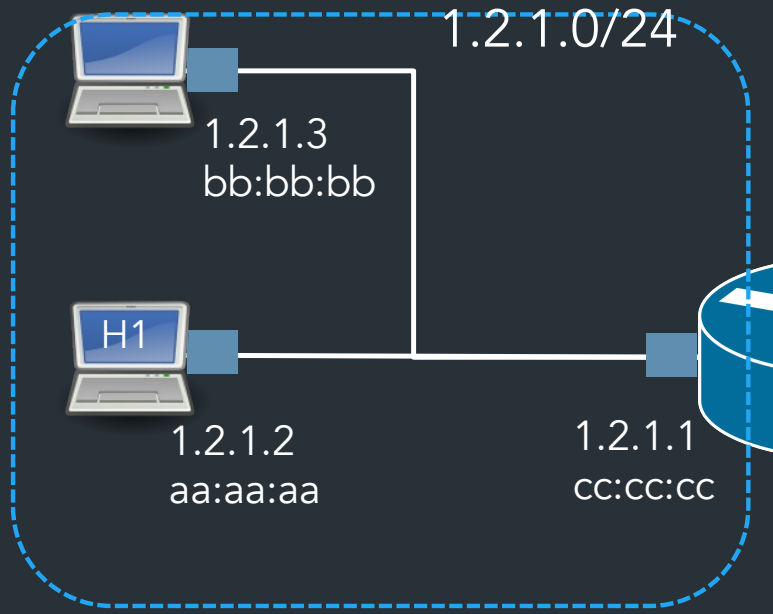
Administrivia

- IP milestone meetings: Should meet with staff today
- IP: Next steps
  - Gearup II: see recording/notes
  - See Implementation Start Guide, other resources
  - Do not leave this project until the last minute

- HW1 due tonight

Warmup

What is the destination MAC address when H1 is sending the following packets?



1)

	Src	Dest
Link	aa:aa:aa	??? ← CC:CC:CC
IP	1.2.1.2	1.2.1.1

2)

	Src	Dest
Link	aa:aa:aa	??? BB:BB:BB
IP	1.2.1.2	1.2.1.3

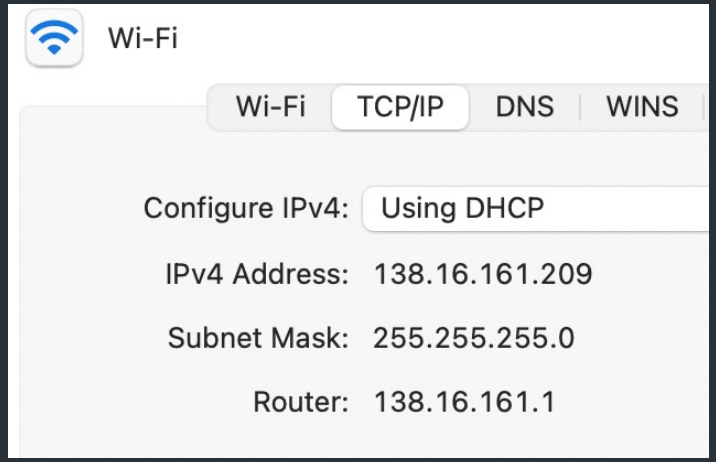
H1's forwarding table:

Prefix	IF/Next hop
1.2.1.0/24	IF1
0.0.0.0	1.2.1.1

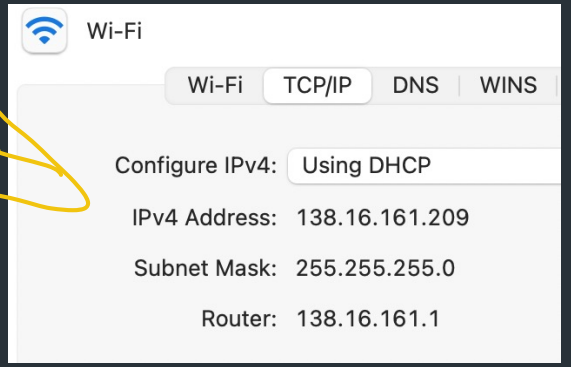
3)

	Src	Dest
Link	aa:aa:aa	??? CC:CC:CC ← NEXT HOP
IP	1.2.1.2	8.8.8.8 (Google) ← FINAL DEST OF PACKET

How do you get an IP address?



Getting an IP



Two ways to configure an IP for a host:

- Static configuration: manually specify IP address, mask, gateway, … Only use this for devices that don't change often

- Automatic: ask the network for an IP when you connect!

  => More common for end hosts
  => DHCP: Dynamic Host Configuration Protocol (end hosts, home routers)

Getting an IP

Two ways to configure an IP for a host:

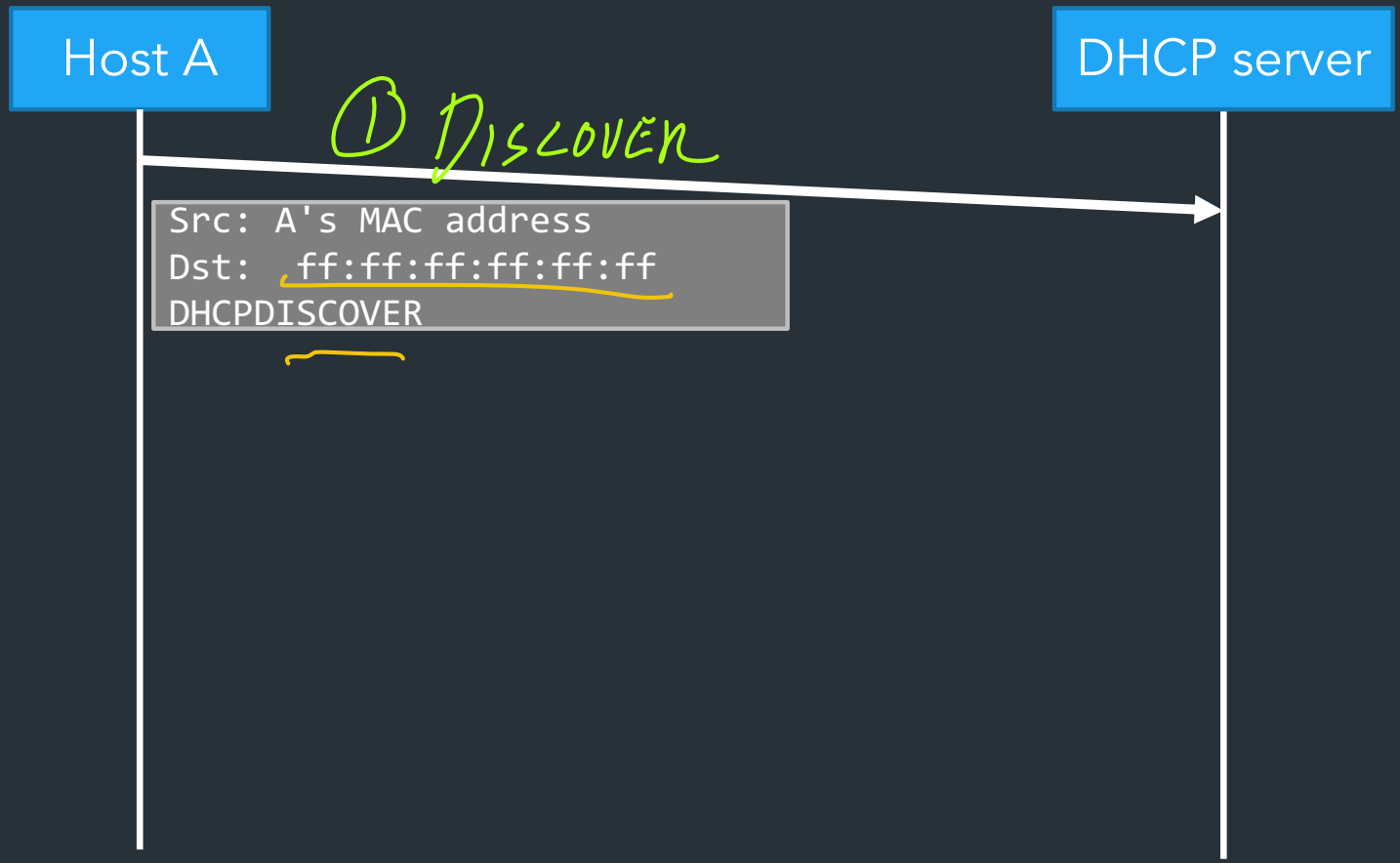
- Static configuration: manually specify IP address, mask, gateway, …

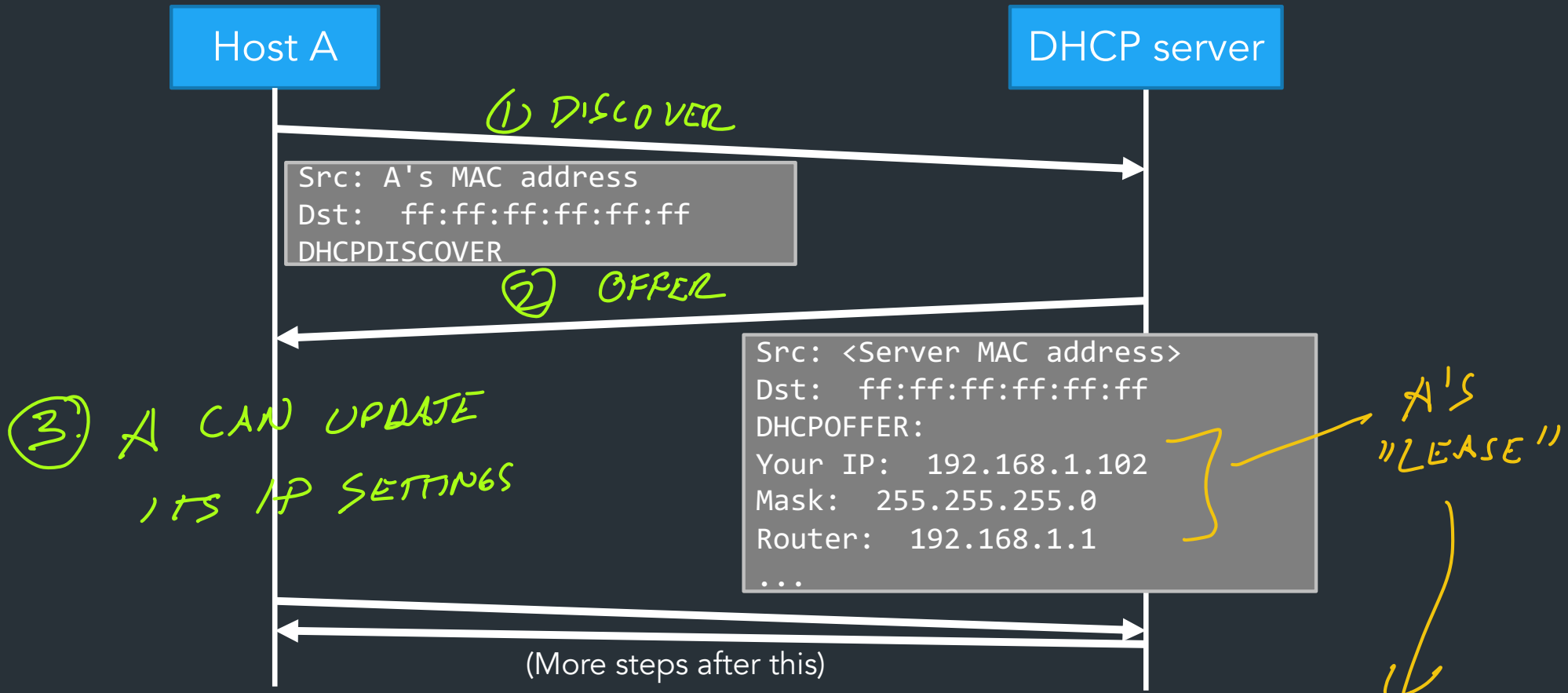
  => More common with network devices that don't change often

- Automatic: ask the network for an IP when you connect!

  => Most common for end hosts

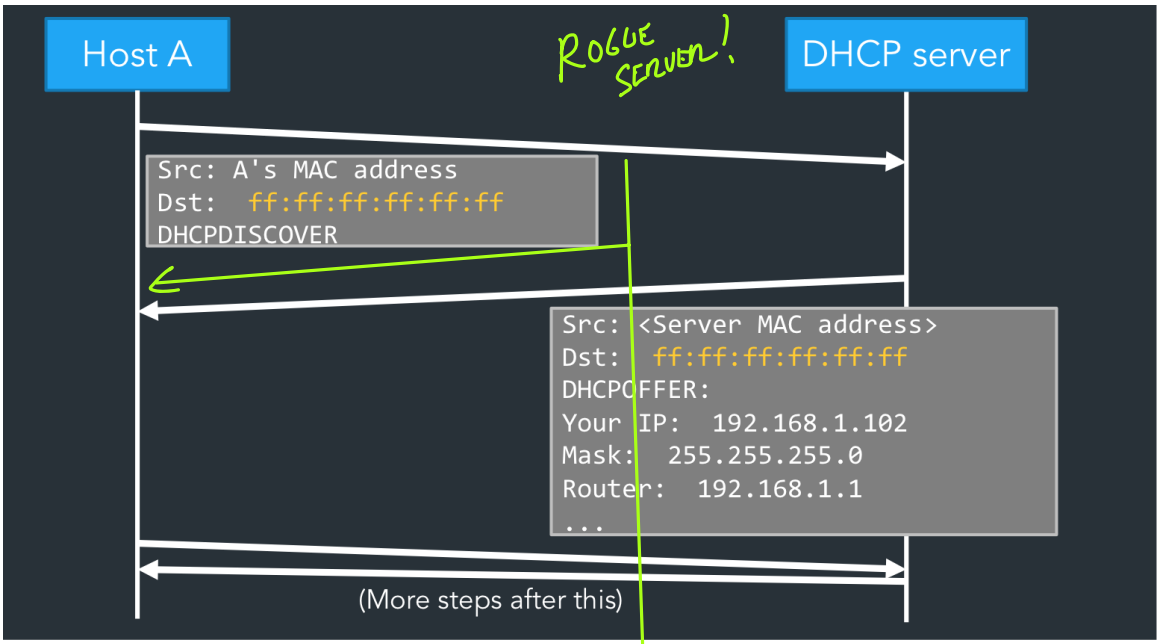
  => Dynamic Host Configuration Protocol (DHCP)


Host A
DHCP server
① Discover
Src: A's MAC address
Dst: ff:ff:ff:ff:ff:ff
DHCPDISCOVER



The server's offer provides enough info for A to configure its IP settings (eg. address, mask, default gateway, + other info we'll discuss later
=> Known as "DHCP options"

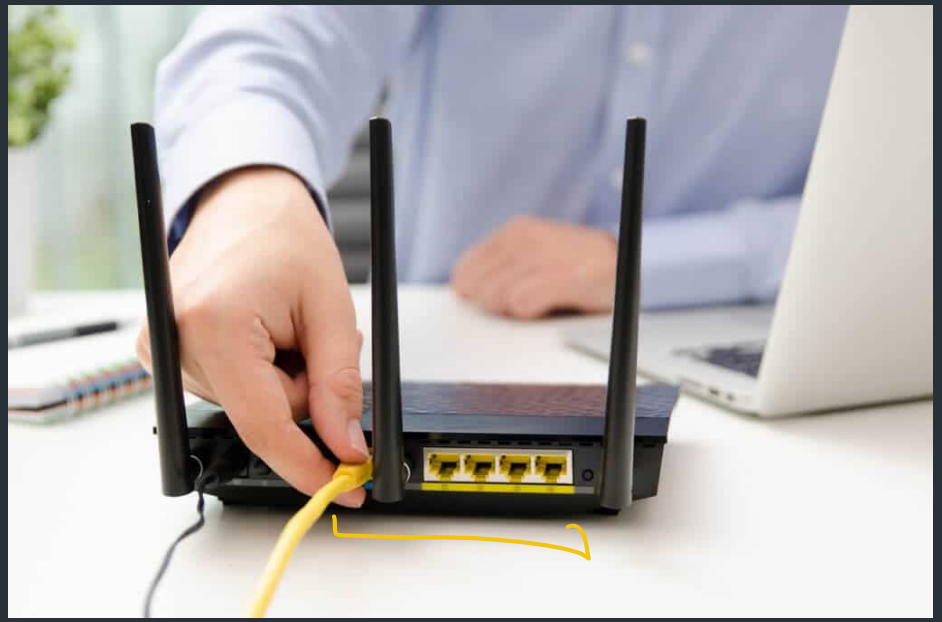
DHCP: What can go wrong?



Problem: just like with ARP, DHCP requests are sent to a broadcast address, so in theory any device can pretend to be the DHCP server and respond.

This would cause hosts to have incorrect settings, and could be a security risk (eg. the DHCP server could even configure the host's settings to intercept its traffic)!

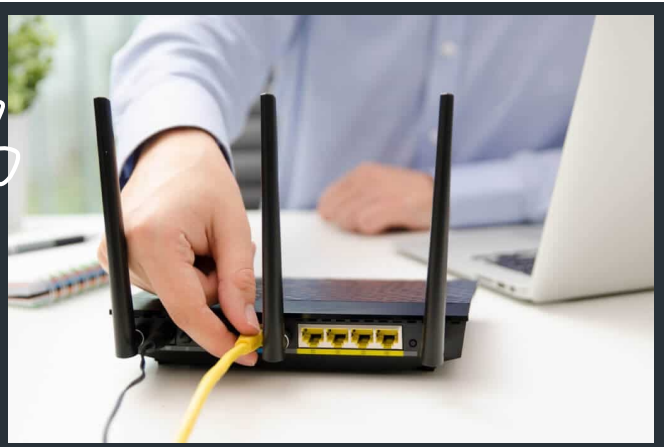
However, this problem is often detectable: network devices and other DHCP servers can monitor DHCP traffic, and could detect when a) a rogue server is detected handing out leases, b) someone is using an IP that wasn't given out as a lease from a valid DHCP server.

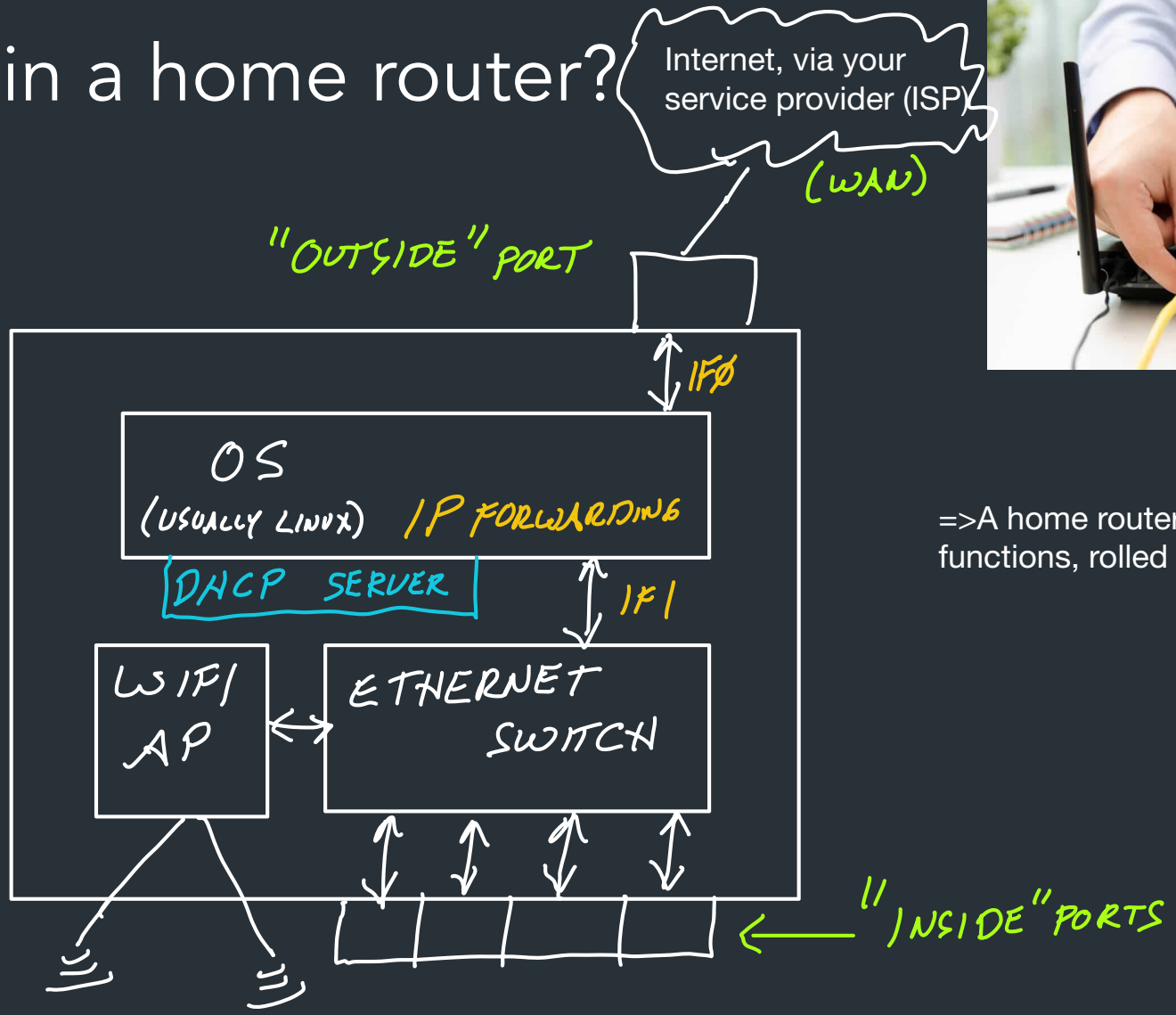


Home routers

The good, the bad, and the ugly…

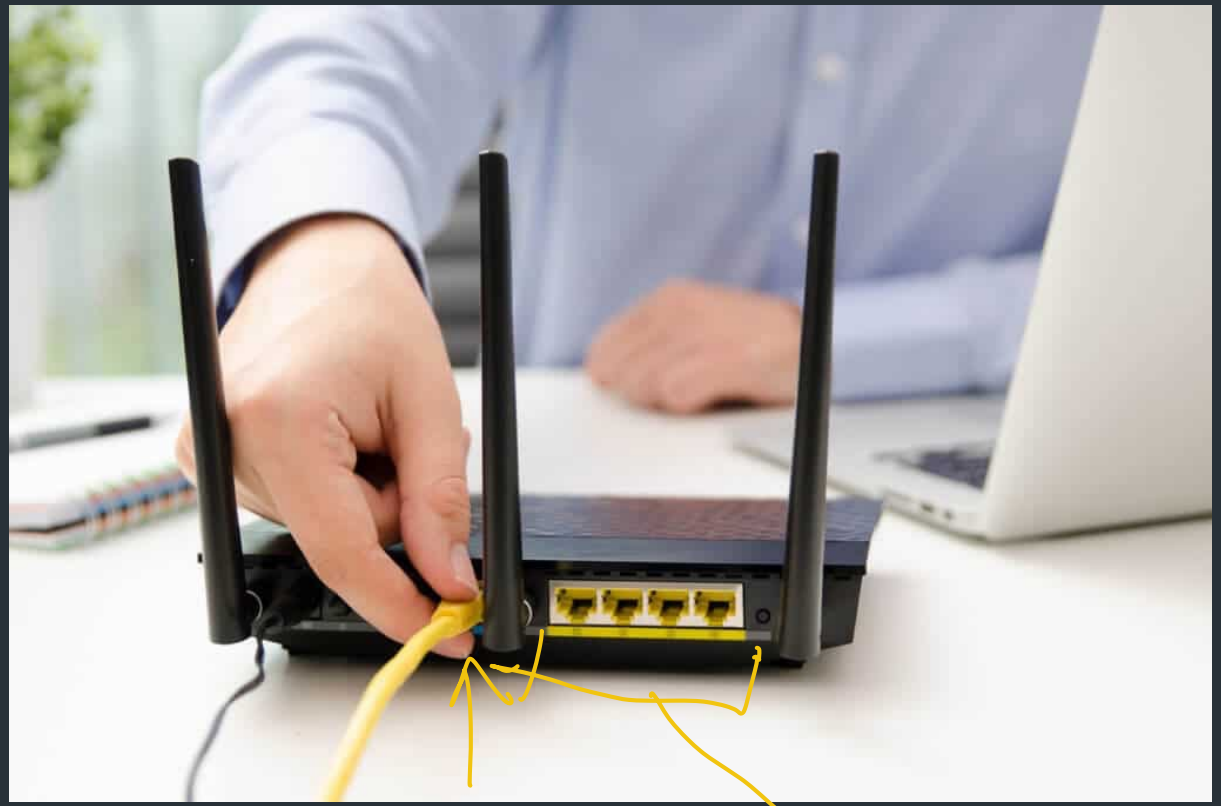
What's in a home router?





=>A home router performs all of these functions, rolled into one device!

Story time



Recap: IP vs. Link-layer address

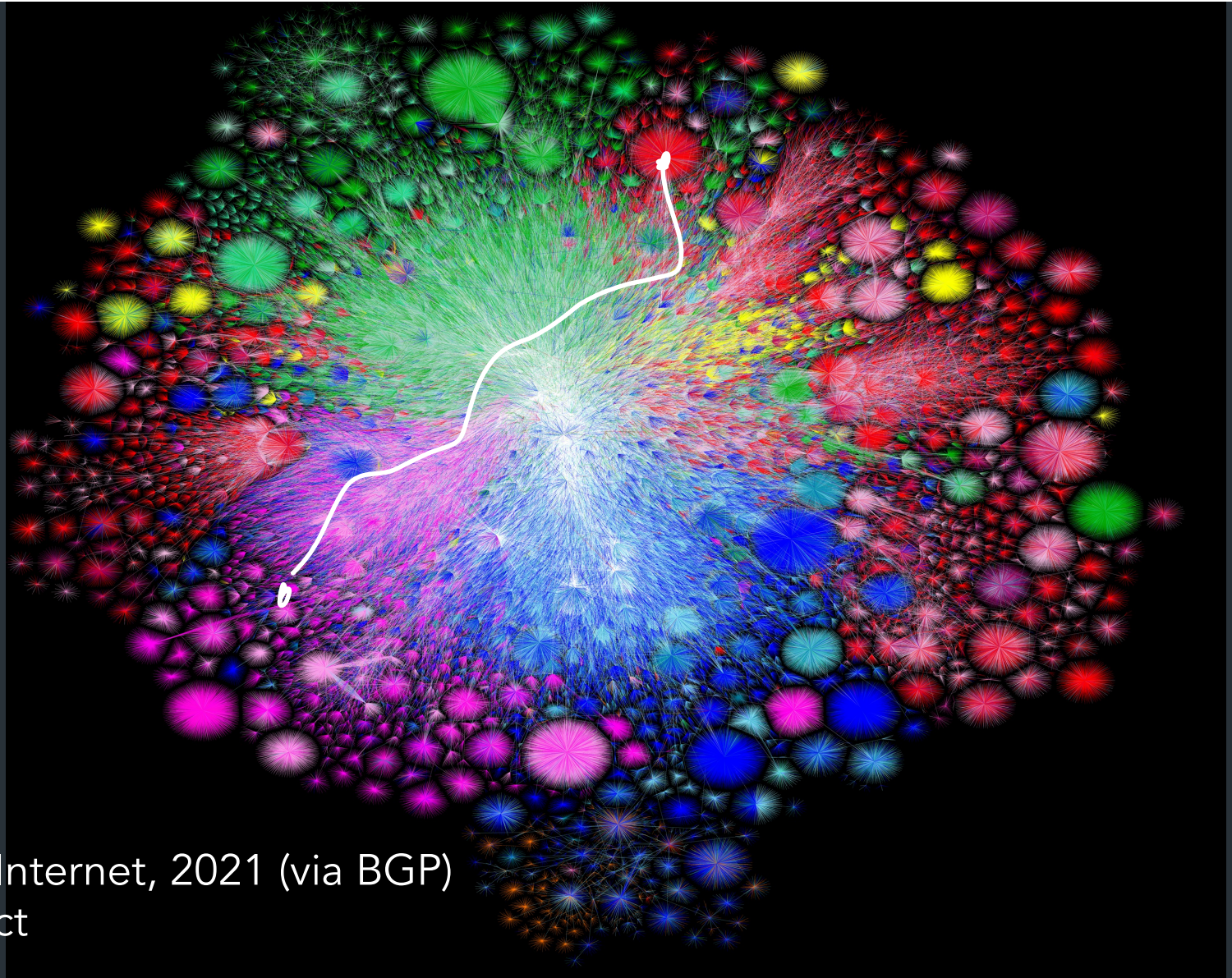
	Src	Dest
Link	aa:aa:aa	cc:cc:cc
IP	1.2.1.2	8.8.8.8 (Google)

<u>Link-layer header info (Ethernet/Wifi/etc)</u>

- Destination MAC address is link-layer addr for packet's next hop
- Changes every hop
- Each hop could use a different link-layer protocol!

<u>IP header info</u>

- Destination IP is IP address of packet's final destination
- Routers look at destination IP to figure out where packet goes next (and which MAC address goes on packet next)



Map of the Internet, 2021 (via BGP)
OPTE project

Routing

Challenges in moving packets

- Forwarding:

  given a packet, decide which interface to send the packet (based on destination IP)

  => Occurs on every packet

- Routing:

  network-wide process of determining the packet's *path* => how routers figure out what to put in their forwarding tables

  => figuring out how to keep table updated as network changes

  => Slower process, not per packet (many seconds, minutes)

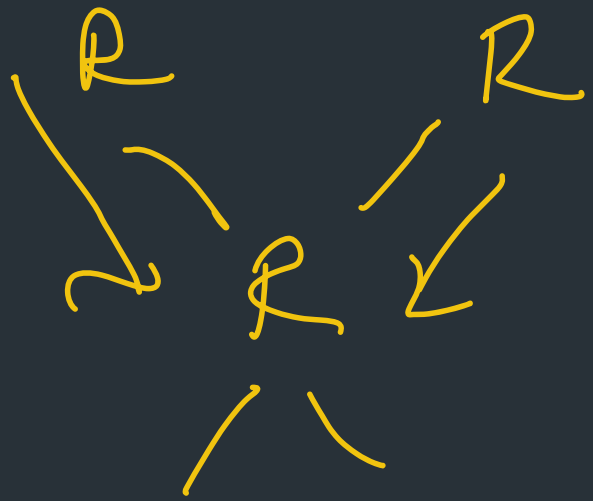
Routing is the process of updating forwarding tables

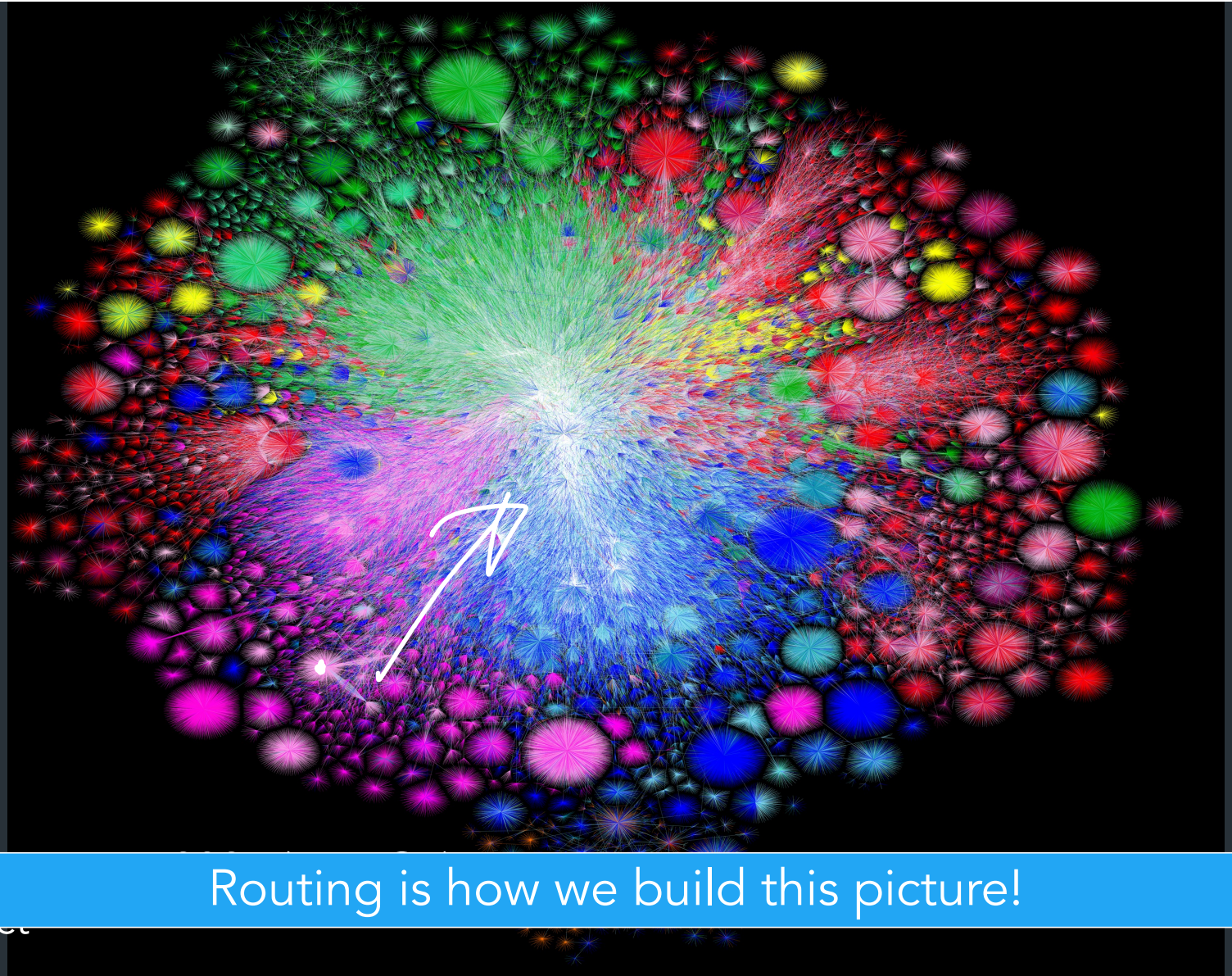
- Routers exchange messages about networks they can reach

Goal: find optimal route (or *any* route…) for every other destination

This is a hard problem

- Decentralized
- Topology always changing
- Scale!





Map of th
OPTE proj

Routing is how we build this picture!

How do we connect *everything*?

Relies on hierarchical nature of IP addressing

- Smaller routers don't need to know everything, just another router that knows more

  ⇒ Has default route

- Core routers know everything => no default!

A forwarding table (my laptop)

0.0.0.0/0

```
deemer@ceres ~ % ip route
default via 10.3.128.1 dev wlp2s0
10.3.128.0/18 dev wlp2s0 proto dhcp scope link src 10.3.135.44 metric 3003
172.18.0.0/16 dev docker0 proto kernel scope link src 172.18.0.1
192.168.1.0/24 dev enp0s31f6 proto kernel scope link src 192.168.1.1
```

A large table

```
rviews@route-server.ip.att.net>show route table inet.0 active-path

inet.0: 866991 destinations, 13870153 routes (866991 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both


0.0.0.0/0          *[Static/5] 5w0d 19:43:09
                    > to 12.0.1.1 via em0.0
1.0.0.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 13335 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.4.0/22         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.4.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.5.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
1.0.6.0/24         *[BGP/170] 1d 10:24:47, localpref 100, from 12.122.83.238
                      AS path: 7018 3356 4826 38803 I, validation-state: valid
                    > to 12.0.1.1 via em0.0
```

Thinking about the scale

At this stage, we think about routing to *whole networks*, ie, some entity with some set of IP prefixes:

eg. Brown University @ 128.148.0.0/16, 138.16.0.0/16

We call each entity an Autonomous System (AS):
a single administrative domain that lives on the Internet

Routing is organized in two levels:

- Intra-domain (interior) routing: routing within an AS
  - ~ 100 PREFIXES / ROUTERS
  - ADMINISTRATOR CONTROLS ALL ROUTERS
  - KNOW ABOUT ALL ROUTERS ⇒ CAN TRY TO FIND SHORTEST PATH

  (RIP, OSPF)

  WE ARE HERE

- Inter-domain (exterior) routing: routing between ASes
  - NO SINGLE ADMIN
  - DON'T HAVE ALL INFO.
  - DECISIONS MADE BY POLICY

  (BGP)

  ⇒ INTERNET-SCALE.

Routing is organized in two levels:

(RIP, OSPF)

- Intra-domain (interior) routing: routing within an AS
  - => Full knowledge of the network inside the AS
  - => One administrator, routing policy
  - => Strive for optimal paths

^ We are here today

(BGP)

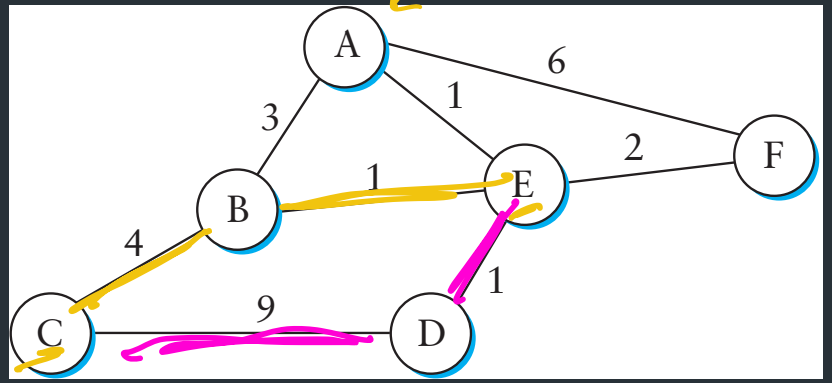
- Inter-domain (exterior) routing: routing between ASes
  - => None of the above, decisions instead made by *policy* (later)

=> INTERNET-SCALE

Intra-Domain (Interior) Routing

Typically, view network as a graph

- Nodes are routers
- Assign some *cost* to each edge
  - latency, b/w, queue length, …



ONE ROUTER

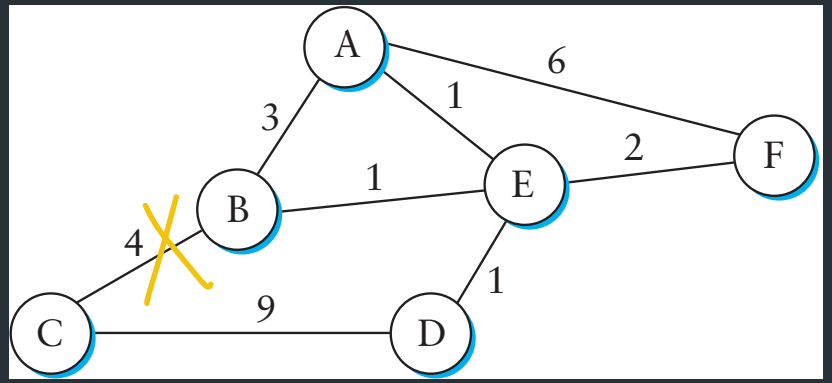
"COST" OR "METRIC" SET BY ADMIN

Goal: find lowest-cost path between nodes

- Each node individually computes routes

Typically, view network as a graph

- Nodes are routers
- Assign some *cost* to each edge
  - latency, b/w, queue length, …



DECENTRALIZED.

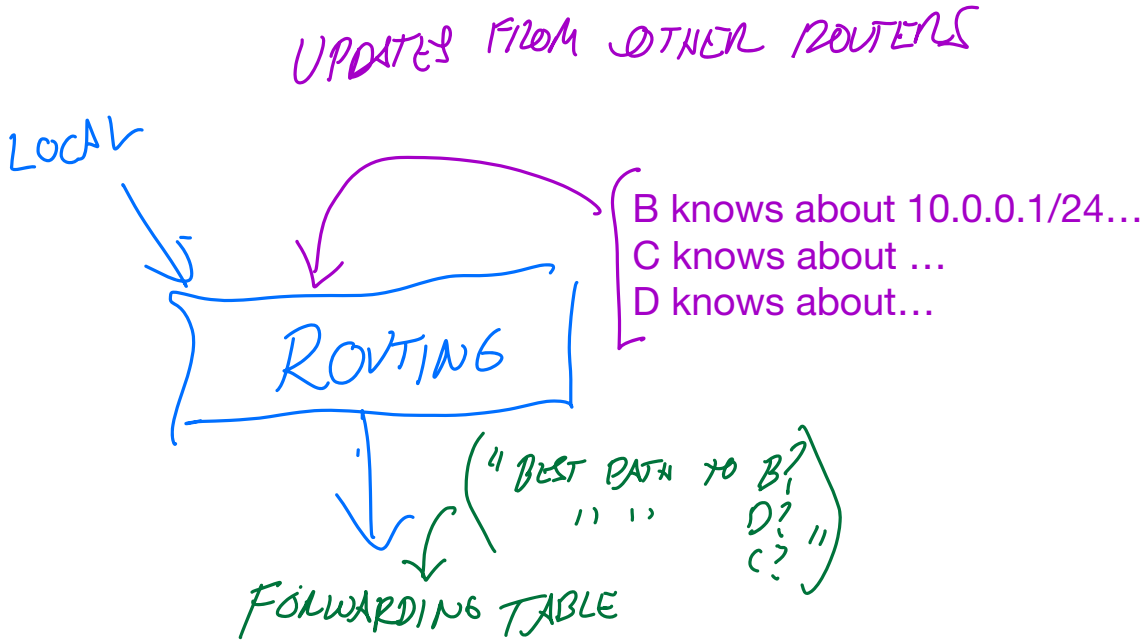
Goal: find lowest-cost path between nodes

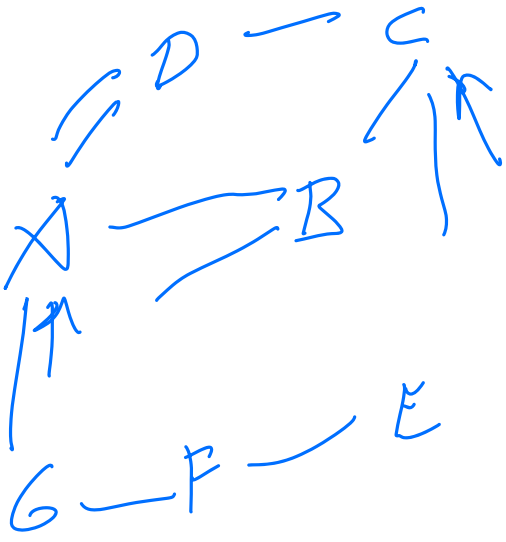
- Each node individually computes routes

Collect routes into a *routing table*, used to generate the forwarding table based on lowest-cost path

Routing process is decentralized:

- No one entity telling routers what routes to have
- When we do interior routing, there is algorithm that routers are using, independently, to figure out what to do





Generally: routing algorithms are *decentralized*

=>In general, no one entity telling routers what routes to use

=> Even for “interior” routing, where there is one admin, routers independently compute how to update their tables based on latest info from other routers

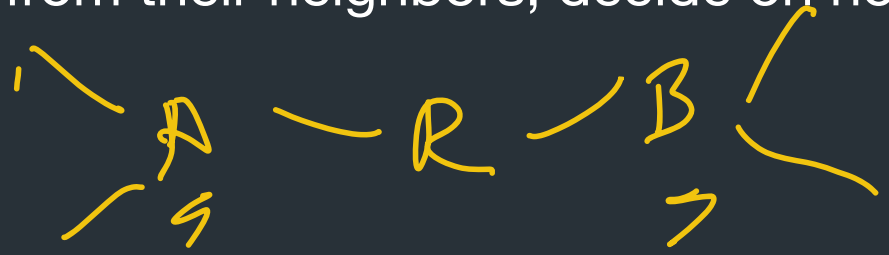
Two classes of intra-domain routing algorithms

TODAY

Distance Vector (Bellman-Ford shortest path algorithm)

RIP

Routers learn info from their neighbors, decide on how to update tables



Link State (Djikstra/Prim shortest path algorithm)

OSPF

Distance Vector Routing

- Each node maintains a *routing table*

- Exchange *updates* with neighbors about node's links:
  => List of <Destination, Cost> pairs

"I KNOW ABOUT A AT COST 2"
(A, 2)

Distance Vector Routing

- Each node maintains a routing table
- Exchange *updates* with neighbors about node's links:
  => List of <Destination, Cost> pairs
- When to send updates?
  - Periodically (seconds to minutes)
  - Whenever table changes (*triggered* update)
  - Time out an entry if no updates within some time interval

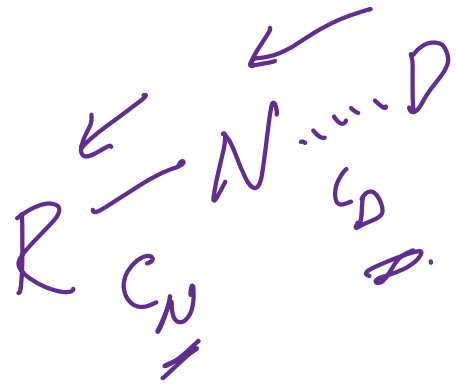
Distance Vector Routing

- Each node maintains a routing table
- Exchange *updates* with neighbors about node's links:
  => List of <Destination, Cost> pairs
- When to send updates?
  - Periodically (seconds to minutes)
  - Whenever table changes (*triggered* update)
  - Time out an entry if no updates within some time interval

Dest.	Cost	Next Hop
A	3	S
B	4	T
C	5	S
D	6	U

DV routing: update rules

Suppose: router receives an update about some destination D from neighbor router N



(D, C_D)

$\Rightarrow$ R CAN REACH D WITH COST $C = C_D + C_N$

(D, C_D)

Table has format <Destination, Cost, Next hop>

If D isn't already in the table, add it <D, C, N>

If you have an existing entry <D, c_old, M>

- c < c_old => Update table with new route <D, c, N> (BETTER ROUTE!)

- if c > c_old {
  if (N == M) // Topology has changed, route has higher cost now (HIGHER COST!)
  Update your table: <D, c, N>

  else // N != M
  // Can ignore (current route is better!)
  }
- c == c_old and N == M
  // Repeat of same route => No change (but refresh timeout)

Separately: need to keep track of last update time for each route, delete old entries that expire

Distance Vector: Update rules

Say router R receives an update <D, c_D> from neighbor N at cost C_N

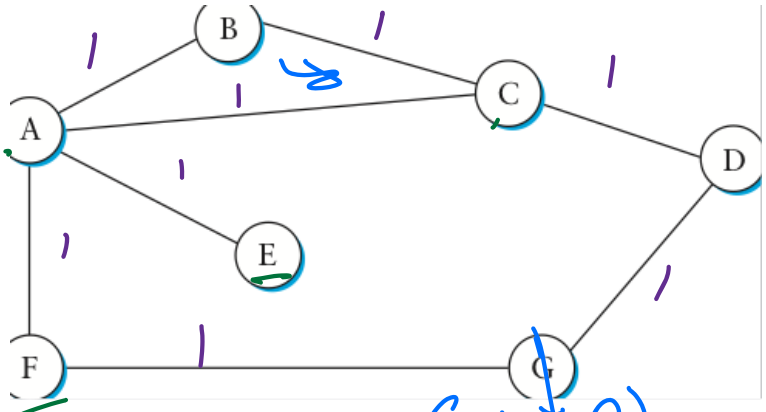
=> Know: R can reach D via N with cost c = c_D + c_N

How to update table?

1. If D not in table, add <D, c, N> (New route!)
2. If table has entry <D, M, c_{old}>:
   - if c < c_{old}: update table to <D, c, M>. (Lower cost!)
   - if c > c_{old} and M == N: update table to <D, c, N> (Cost increased!)
   - if c > c_{old} and M != N: ignore (N is better)
   - if c == c_{old} and M ==N: no change (No new info)
     (Just refresh timeout)

DEST	COST	NEXT
B	0	B *
A	1	A
C	1	C
E	2	A
F	2	A

*: Router always knows about its local prefixes (don't usually write this)



TO: B SENDS UPDATE (B, 0) TO A, C

A SENDS UPDATE (A, 0)

C " " (C, 0)

T_1: — A SENDS UPDATE (A, 0)

(E, 1) NEW!

(F, 1) NEW!

(C, 1)] NOT BETTER!

(B, 1)]

Warmup

Suppose router R has the following table:

Dest.	Cost	Next Hop
A	3	S
B	4	T
C	~~5~~ 6	S
D	~~6~~ 5	~~U~~ S

—NO CHANGE

—TIE – COULD PICK EITHER

—COST TO C FROM S INCREASED BY 1 ⇒ MUST UPDATE!

S HAS BETTER ROUTE TO D ⇒ UPDATE!

E 3 S

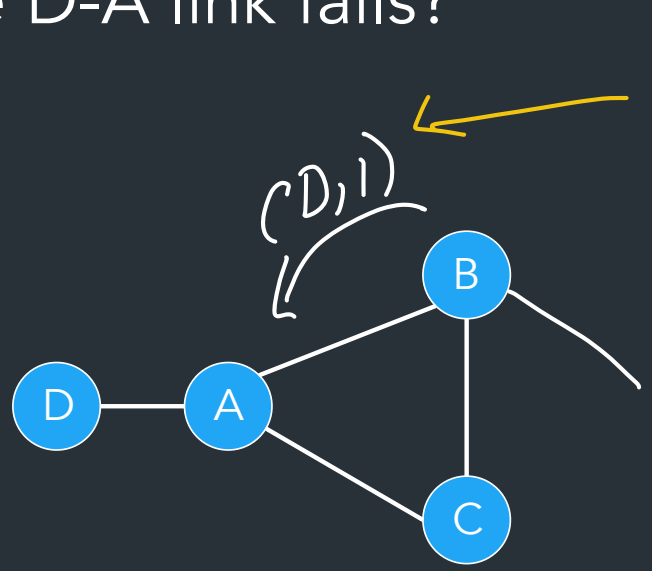
What happens when it gets this update from router S?

Dest.	Cost
A	2
B	3
C	5
D	4
E	2

← NEW!

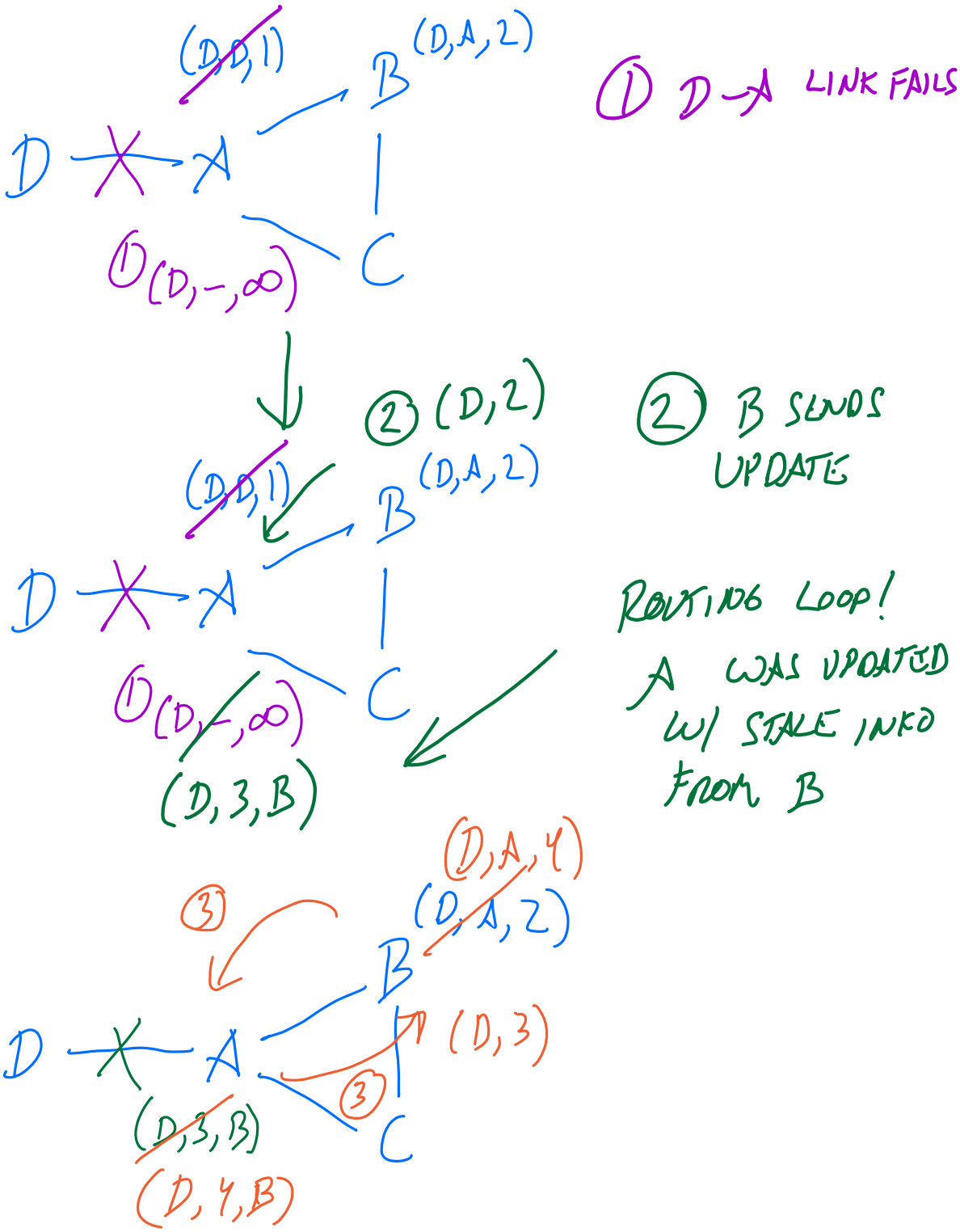
After this page are more notes from the next lecture from last year
=> Feel free to read ahead!

What happens when the D-A link fails?



ONLY INFO CONTAINED IN A DISTANCE-VECTOR UPDATE!

Updates occur in a loop with increasing cost until cost reaches infinity (16)!
=> Count to infinity => long time to converge when links fail



Count to infinity: cost keeps increasing until it reaches infinity

=> "Bad news travels slowly"

=> In RIP: "infinity" == 16

Why does this happen? DV only based on info from neighbors, and not enough info to resolve loops, etc.

Can we avoid loops?

- Does IP TTL help? Nope.
- Simple approach: consider a small cost *n* (e.g., 16) to be infinity

Fundamental problem: distance vector only based on local information!
=> Not enough info to resolve loops, race conditions, count-to-infinity,
but there are some tricks…

RFC1058 (1988): The original RIP standard*

supply the information that is needed to do routing.

1.1. Limitations of the protocol

This protocol does not solve every possible routing problem. As mentioned above, it is primary intended for use as an IGP, in reasonably homogeneous networks of moderate size. In addition, the following specific limitations should be mentioned:

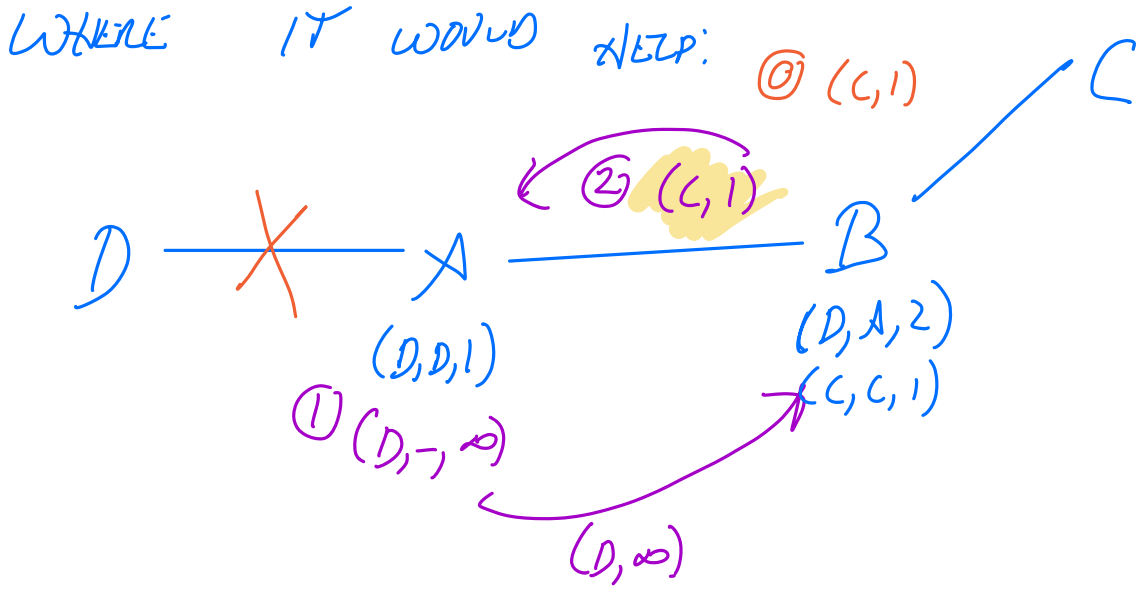
*: Obsoleted by RFC2453 (don't use RFC 1058 for the project, Use RFC 2453 instead)

One strategy: Split Horizon

- When sending updates to node A, don't include routes you learned from A
- Prevents B and C from sending cost 2 to A

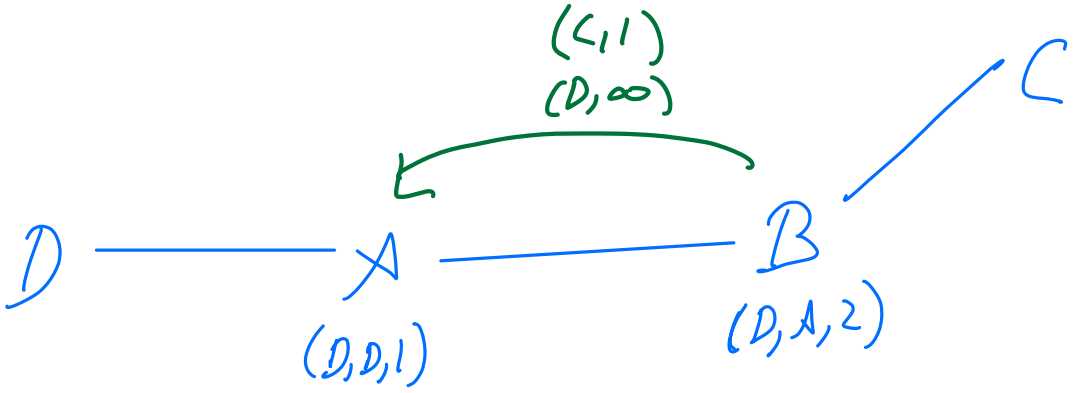
A solution (at least for RIP): **Split Horizon**

Definition: If A uses N as next hop for D, do not report to N about D
=> Prevents "linaer" routing loops, but not others




```
What happened?
  1) D-A link fails
  2) B's updates to A don't include any info
about D => no change to A's table (wrt D)
  3) A updates B => (D, inf)
```

Commonly used with: **Poison reverse**: rather than not including routes learned from A, explicitly send cost of infinity
=> Idea: may help converge in some cases (but hard to see it in practice)



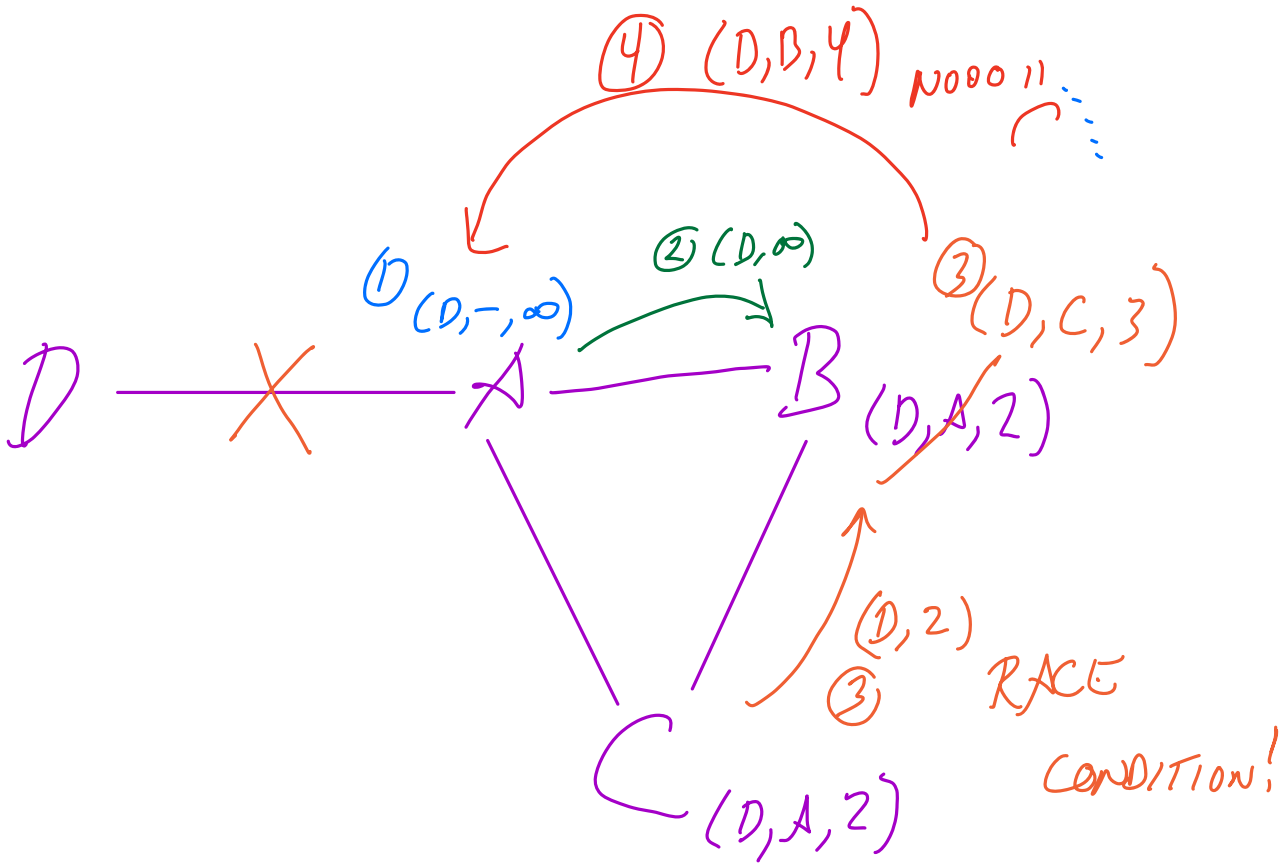
Split Horizon + Poison reverse

- Rather than not advertising routes learned from A, explicitly include cost of ∞.
- Faster to break out of loops, but increases advertisement sizes

⇒ Does it help? Not completely.

=> A common convention, might reduce time to converge, but overall hard to see effect vs. split horizon

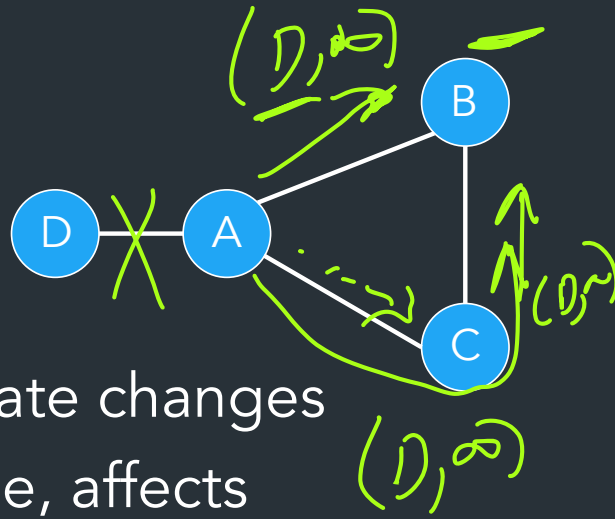
But even this can't prevent all loops!!!



What happens?
1) D-A link fails
2) A updates B => (D, inf)
3) Before C gets the same update, it sends (D, 2) to B
   => RACE CONDITION!!! C might send old update to B before C gets update from A
4) B updates A, overwrites A's table
5) ... count to infinity ...

So what can we do?
- Can't send any extra information.

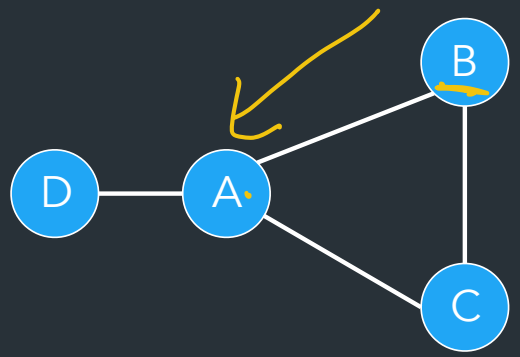
Even with split horizon + poison reverse,
can still create loops with >2 nodes!



What else can we do?

- Triggered updates: send update as soon as link state changes
- Hold down: delay using new routes for certain time, affects convergence time

Practice



B's routing table

Dest.	Cost	Next Hop
A	1	A
C	1	C
D	2	A

Routers A,B,C,D use RIP. When B sends a periodic update to A, what does it send…

- When using standard RIP?
- When using split horizon + poison reverse?

STANDARD
(A, 1)
(C, 1)
(D, 2)

SH + PR
(A, ∞)
(C, 1)
(D, ∞)

From RFC2453, RIP v2 (1998):

3.2 Limitations of the Protocol

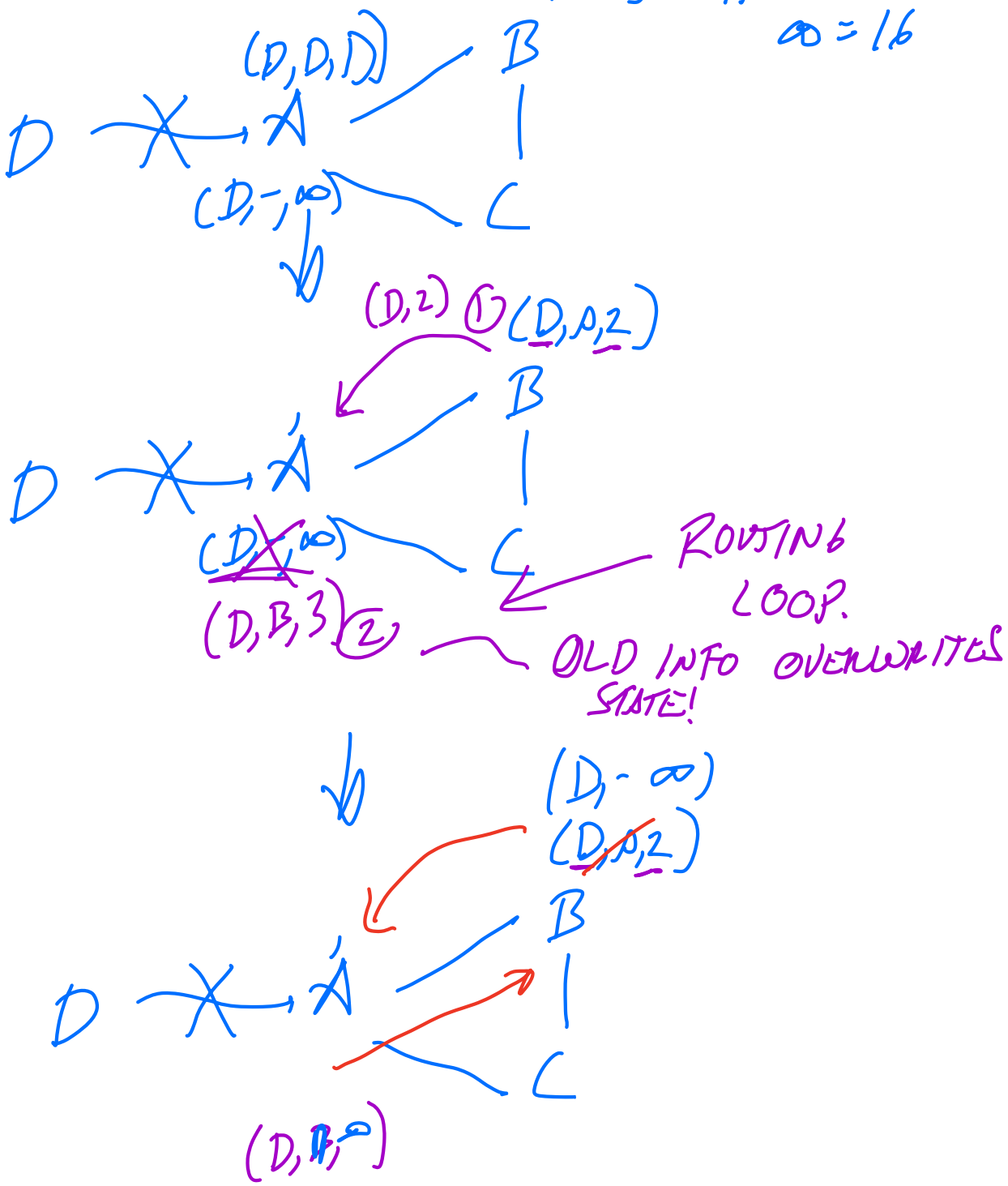
This protocol does not solve every possible routing problem. As mentioned above, it is primary intended for use as an IGP in networks of moderate size. In addition, the following specific limitations are be mentioned:

- The protocol is limited to networks whose longest path (the network's diameter) is 15 hops. The designers believe that the basic protocol design is inappropriate for larger networks. Note that this statement of the limit assumes that a cost of 1 is used for each network. This is the way RIP is normally configured. If the system administrator chooses to use larger costs, the upper bound of 15 can easily become a problem.

- The protocol depends upon "counting to infinity" to resolve certain unusual situations. (This will be explained in the next section.) If the system of networks has several hundred networks, and a routing loop was formed involving all of them, the resolution of the loop would require either much time (if the frequency of routing updates were limited) or bandwidth (if updates were sent whenever changes were detected). Such a loop would consume a large

More (even older) notes about RIP
(in case you want to see more examples)

RIP: WHAT HAPPENS WHEN D-A LINK FAILS?



- UPDATES OCCUR IN A LOOP W/ INCREASING COST UNTIL COST REACHES ∞

⇒ COUNT TO INFINITY
- LONG CONVERGE TIME.

How to avoid loops

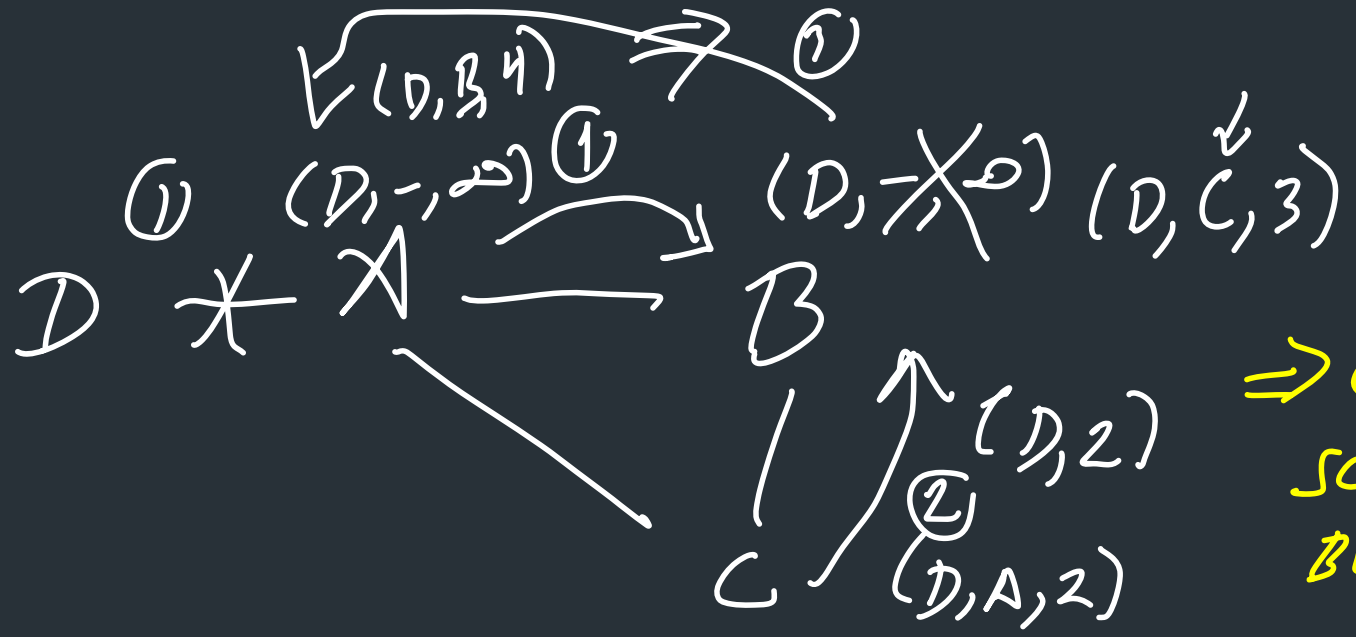
- Does IP TTL help? => doesn't solve routing problem.
- Simple approach: consider a small cost n (e.g., 16) to be infinity
  - After n rounds decide node is unavailable
  - But rounds can be long, this takes time

Problem: distance vector based only on local information

↳ not enough to resolve loops, count-to infinity (but there are tricks we can do)

One way: Split Horizon

- When sending updates to node A, don't include routes you learned from A
- Prevents B and C from sending cost 2 to A

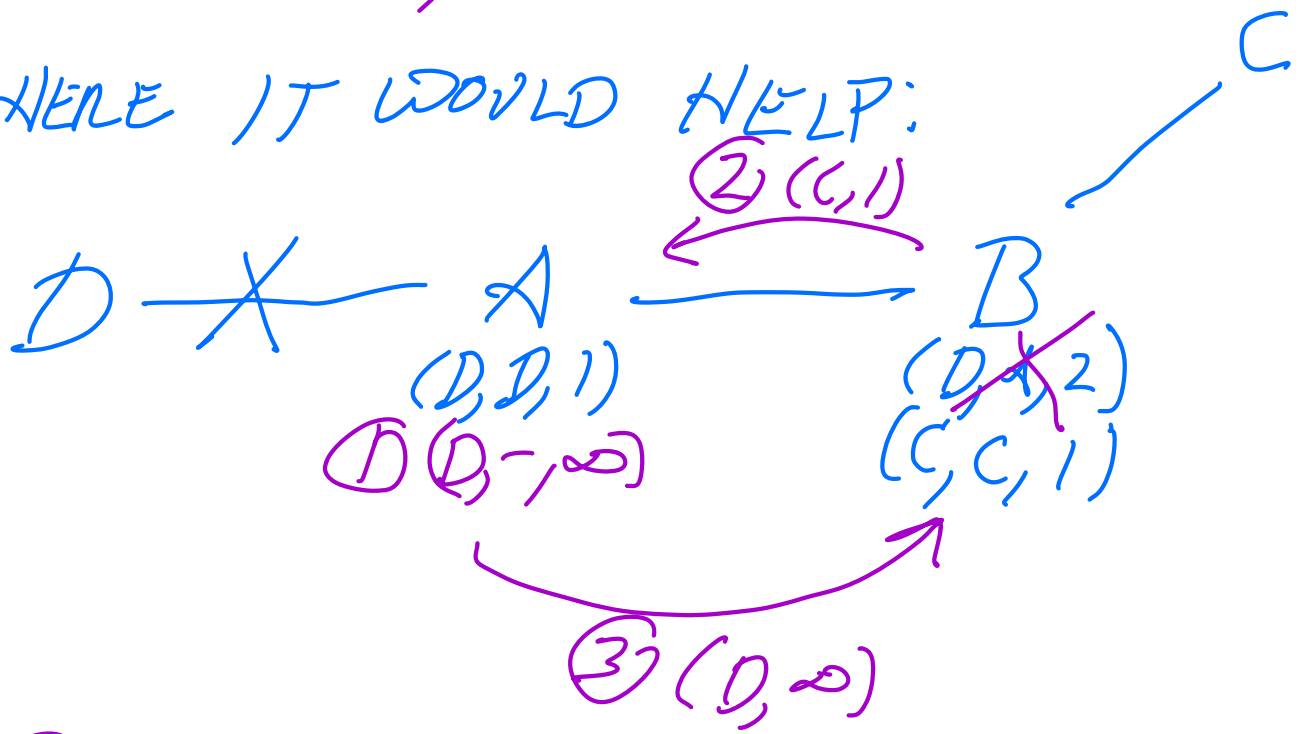


SPLIT HORIZON

- IF A USES N AS NEXT HOP FOR D, DO NOT REPORT TO N ABOUT D

⇒ PREVENTS "LINEAR" ROUTING LOOPS, BUT NOT OTHERS

WHERE IT WOULD HELP:



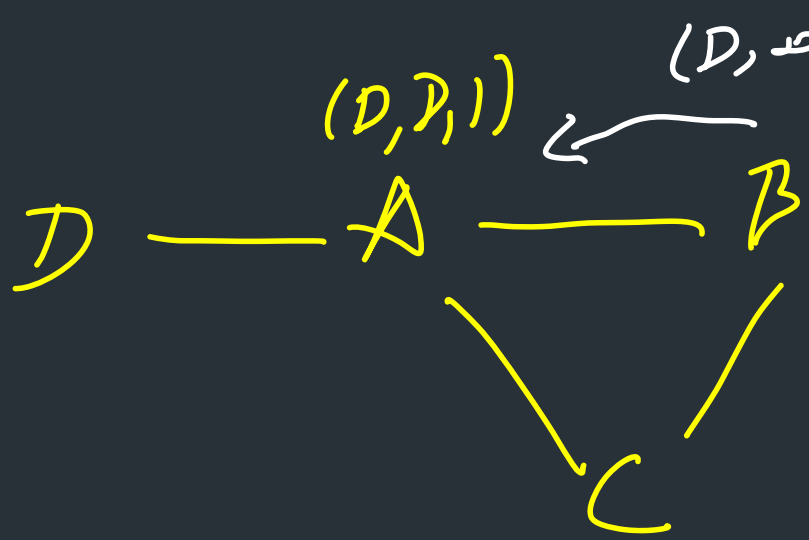
① D-A LINK GOES DOWN

② WHEN B SENDS UPDATE TO A, IT WOULD NOT TELL INCLUDE A

③ A UPDATES B W/ (D, ∞)

Split Horizon + Poison Reverse

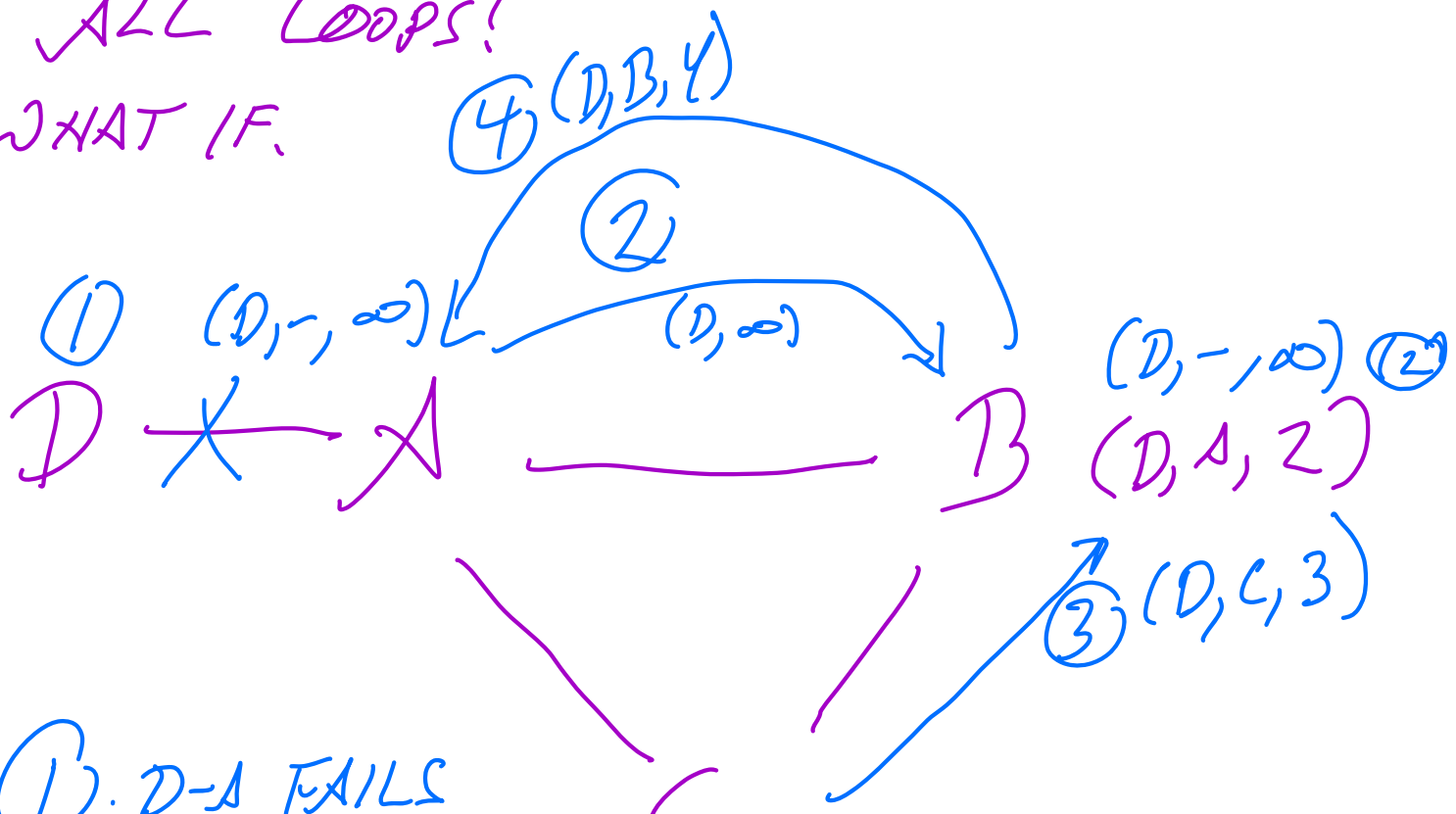
- Rather than not advertising routes learned from A, explicitly include cost of ∞.
- Faster to break out of loops, but increases advertisement sizes



POISON REVERSE IS A COMMON CONVENTION, MIGHT HELP TIME TO CONVERGE IN SOME CASES, BUT HARD TO SEE EFFECT VS. SPLIT HORIZON.

BUT EVEN W/ SPLIT HORIZON + POISON REVERSE, CAN'T PREVENT ALL LOOPS!

WHAT IF.

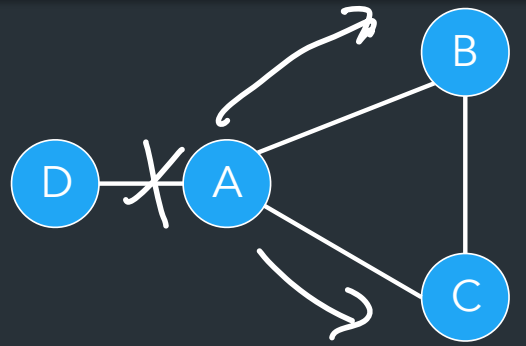


① D-A FAILS
② A UPDATES B (D, ∞)
③ C SENDS (D, 2) TO B!
↳ RACE CONDITION! C MIGHT SEND OLD UPDATE TO B BEFORE C GETS UPDATE FROM A!
④ B UPDATES A, OVERWRITES A'S ENTRY
⑤ ... COUNT TO INFINITY...

WHAT CAN WE DO?

Distance-vector updates

Even with split horizon + poison reverse,
can still create loops with >2 nodes

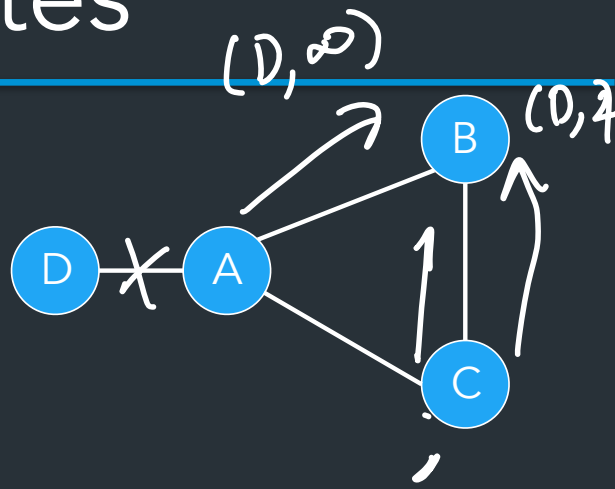


What else can we do? TRIGGERED UPDATES.

- IF A TELLS B AND C IMMEDIATELY THAT ITS ROUTE CHANGES, IT CAN PREVENT THIS LOOP.

Distance-vector updates

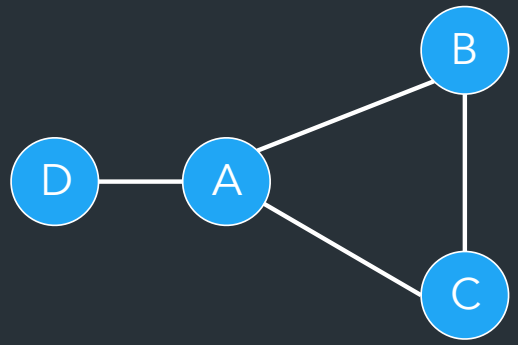
Even with split horizon + poison reverse,
can still create loops with >2 nodes



What else can we do?

- Triggered updates: send update as soon as link state changes
- Hold down: delay using new routes for certain time, affects convergence time

Practice

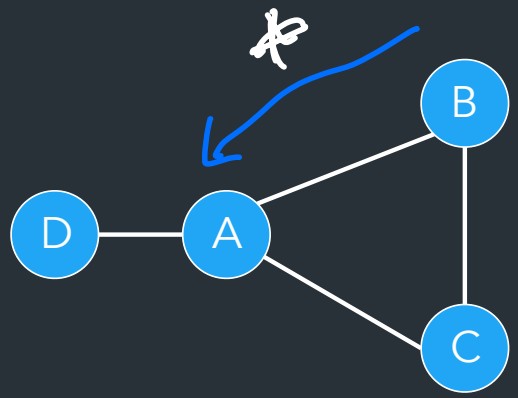


B's routing table

Routers A,B,C,D use RIP. When B sends a periodic update to A, what does it send…

- When using standard RIP?
- When using split horizon + poison reverse?

Practice



B's routing table

Dest.	Cost	Next Hop
A	1 ∞	A
C	1	C
D	2 ∞	A

Routers A,B,C,D use RIP. When B sends a periodic update to A, what does it send…

- When using standard RIP?
- When using split horizon + poison reverse?

STANDARD:
(A, 1)
(C, 1)
(D, 2)

SH + PR
(A, ∞)
(C, 1)
(D, ∞)